

1 < /* SubQuery 개념

2 하나의 쿼리가 다른 쿼리에 포함되는 구조

3 다른 쿼리에 포함된 내부쿼리 (서브쿼리) 는 외부쿼리 (메인 쿼리) 사용될 값을 반환하는 역할

4

5 - WHERE, HAVING 절에서 조건으로 => 단일행/다중행 서브쿼리

6 - FROM 절에서 테이블처럼 => 인라인 뷰 (Inline View)

7 - SELECT 절에서 컬럼처럼 => 스칼라 서브쿼리

8

9 <종류/사용 연산자>

10 - 단일행서브쿼리 (단일열, 다중열) : =, <, >

11 - 다중행서브쿼리 (단일열, 다중열) : IN, ANY, ALL

12

13 단일행과 다중행에서 사용할 수 있는 연산자가 다르다.

14 ** /

15

16

17 -- 나승원 사원의 부서번호를 조회한다면?

18

19 **SELECT** EMP_NAME, DEPT_ID

20 **FROM** EMPLOYEE

21 **WHERE** EMP_NAME = '나승원' ;

22

23 -- 나승원이 속한 부서직원의 정보를 조회하라.

24

25 **SELECT** *

26 **FROM** EMPLOYEE

27 **WHERE** DEPT_ID = (**SELECT** EMP_NAME, DEPT_ID

28 **FROM** EMPLOYEE

29 **WHERE** EMP_NAME = '나승원') ;

30

31

32 /* 3. "국어국문학과" 에 다니는 여학생 중 현재 휴학중인 여학생을 찾아달라는 요청이

33 들어왔다. 누구인가? (국문학과와 '학과코드'는 학과 테이블 (TB_DEPARTMENT) 을 조회해서

34 찾아 내도록 하자) */

35

36

37 **SELECT** DEPARTMENT_NO

38 **FROM** TB_DEPARTMENT

39 **WHERE** DEPARTMENT_NAME = '국어국문학과' ;

40

41 **SELECT** STUDENT_SSN

42 **FROM** TB_STUDENT ;

43

44 **SELECT** STUDENT_NAME

45 **FROM** TB_STUDENT

46 **WHERE** DEPARTMENT_NO = (**SELECT** DEPARTMENT_NO

47 **FROM** TB_DEPARTMENT

48 **WHERE** DEPARTMENT_NAME = '국어국문학과') **AND** ABSENCE_YN = 'Y'

49

```

50 -- 나승원과 같은 직업이고 나승원보다 급여가 많은 사원의 정보를 조회하라.
51
52 SELECT *
53 FROM EMPLOYEE
54 WHERE EMP_NAME = '나승원' ;
55
56 SELECT *
57 FROM EMPLOYEE
58 WHERE JOB_ID = (SELECT JOB_ID,
59                 FROM EMPLOYEE
60                 WHERE EMP_NAME = '나승원')
61 AND
62 SALARY > (SELECT SALARY
63           FROM EMPLOYEE
64           WHERE EMP_NAME = '나승원') ;
65
66 -- 최소급여를 받는 사원 정보를 조회하라...
67
68
69 SELECT MIN(SALARY)
70 FROM EMPLOYEE ;
71
72 SELECT *
73 FROM EMPLOYEE
74 WHERE SALARY = (SELECT MIN(SALARY)
75                 FROM EMPLOYEE) ;
76
77 -- 부서별 급여 총합
78
79 SELECT DEPT_ID, SUM(SALARY)
80 FROM EMPLOYEE
81 GROUP BY DEPT_ID ;
82
83
84 -- 부서별 급여 총합이 가장 높은 부서의 이름과 급여총합을 조회하라.
85
86 SELECT MAX(SUM(SALARY))
87 FROM EMPLOYEE
88 GROUP BY DEPT_ID ;
89
90
91 SELECT DEPT_NAME, SUM(SALARY)
92 FROM EMPLOYEE
93 JOIN DEPARTMENT USING (DEPT_ID)
94 GROUP BY DEPT_ID, DEPT_NAME
95 HAVING SUM(SALARY) = (SELECT MAX(SUM(SALARY))
96                      FROM EMPLOYEE
97                      GROUP BY DEPT_ID) ;

```

```

98< -- 05 03 06
99 -- 부서별 최소급여를 받는 사원을 조회하라...
100
101 -- 부서별 최소급여 (SUBQUERY) > 다중행함수
102 SELECT DEPT_ID, MIN(SALARY)
103 FROM EMPLOYEE
104 GROUP BY DEPT_ID;
105
106 -- 다중행 서브쿼리에서 사용할 수 있는 연산자 (IN, ANY, ALL)
107 -- 다중서브쿼리는 범위로 비교
108 -- IN :EQUALS 비교
109 -- < ANY , > ANY, < ALL, > ALL
110
111 -- 부서별 최소급여를 받는 사원을 조회하라.
112 -- 사원 정보 (NVL: 널값을 다른 값으로 대체) Q
113 SELECT *
114 FROM EMPLOYEE
115 WHERE (NVL(DEPT_ID, ' '), SALARY) IN (SELECT NVL(DEPT_ID, ' ')
116      MIN(SALARY)
117      FROM EMPLOYEE
118      GROUP BY DEPT_ID);
119
120
121
122 /*과장보다 많은 급여를 받는 대리의 급여를 조회*/
123
124 /*대리 급여*/
125 SELECT SALARY
126 FROM EMPLOYEE E
127 JOIN JOB J ON (E.JOB_ID = J.JOB_ID)
128 WHERE JOB_TITLE = '대리';
129
130 /*과장 급여*/
131 SELECT SALARY
132 FROM EMPLOYEE E
133 JOIN JOB J ON (E.JOB_ID = J.JOB_ID)
134 WHERE JOB_TITLE = '과장';
135
136
137 SELECT JOB_TITLE, SALARY
138 FROM EMPLOYEE E
139 JOIN JOB J ON (E.JOB_ID = J.JOB_ID)
140 WHERE JOB_TITLE = '대리'
141 AND SALARY > ALL (SELECT SALARY
142      FROM EMPLOYEE E
143      JOIN JOB J ON (E.JOB_ID = J.JOB_ID)
144      WHERE JOB_TITLE = '과장');
145

```

```

146
147 -- 상관관계 SUBQUERY (Correlated Subquery)
148 -- MAIN 실행 -> SUBQUERY 실행 -> MAIN 실행
149
150 -- EXISTS: 서브쿼리가 결과를 하나라도 반환하면 TRUE
151 -- NOT EXISTS: 서브쿼리가 결과를 하나도 반환하지 않으면 TRUE
152
153 -- 관리자를 구하라.
154
155 SELECT EMP_ID, EMP_NAME, '관리자' AS "구분"
156 FROM EMPLOYEE E
157 WHERE EXISTS ( SELECT NULL
158                FROM EMPLOYEE
159                WHERE E.EMP_ID = MGR_ID)
160 UNION
161 SELECT EMP_ID, EMP_NAME, '직원' AS "구분"
162 FROM EMPLOYEE E
163 WHERE NOT EXISTS ( SELECT NULL
164                    FROM EMPLOYEE
165                    WHERE E.EMP_ID = MGR_ID)
166 ORDER BY 3;
167
168
169 -- SET OPERATOR: SET 연산자
170 -- 두 개 이상의 쿼리 결과를 하나로 결합시키는 연산자. 이질적인 데이터를 하나의 데이터로
171 -- 주의사항) 컬럼의 숫자가 일치, 각 컬럼의 타입이 일치, 컬럼의 숫자가 일치하지 않을 때
172
173 -- <종류>
174 -- UNION: 합집합 (중복제거)
175 -- UNION ALL: 합집합 (중복허용)
176 -- INTERSECT: 교집합
177 -- MINUS: 차집합 (Oracle에서 사용)
178 -- EXCEPT: 차집합 (PostgreSQL 등에서 사용)
179
180 SELECT EMP_ID,
181        ROLE_NAME
182 FROM EMPLOYEE_ROLE
183 MINUS
184 SELECT EMP_ID,
185        ROLE_NAME
186 FROM ROLE_HISTORY;
187
188 -- INLINE VIEW: FROM절에서 명시되는 서브쿼리. VIRTUAL TABLE
189
190 -- 직급별 평균급여를 조회하라
191
192 SELECT JOB_ID, TRUNC(AVG(SALARY), -5) AS JOBAVG /*TRUNC (절삭) :
193 FROM EMPLOYEE
194 GROUP BY JOB_ID;

```

```

195<
196
197--Q
198-- 직급별 평균급여를 받는 사원의 사원정보를 조회하고 싶다면?
199
200 SELECT EMP_NAME,
201        JOB_TITLE,
202        SALARY
203 FROM ( SELECT JOB_ID, TRUNC(AVG(SALARY),-5) AS
204        FROM EMPLOYEE
205        GROUP BY JOB_ID ) V
206 JOIN EMPLOYEE E ON (V.JOB_AVG = E.SALARY
207        AND
208        NVL(V.JOB_ID, ' ') = NVL(E.JOB_ID, ' '))
209 LEFT JOIN JOB J ON (E.JOB_ID = J.JOB_ID);
210
211
212
213 /*14. 춘 기술대학교 서반아어학과 학생들의 지도교수를 게시하고자 한다. 학생이름과
214 지도교수 이름을 찾고 만일 지도 교수가 없는 학생일 경우 "지도교수 미지정"으로
215 표시하도록 하는 SQL 문을 작성하시오. 단, 출력헤더는 , 학생이름\, , 지도교수\로
216 표시하며 고향번호 학생이 먼저 표시되도록 한다. */
217 /*다시 풀기*/
218
219
220 SELECT *
221 FROM TB_STUDENT
222 WHERE COACH_PROFESSOR_NO IS NULL;
223
224 SELECT DEPARTMENT_NO
225 FROM TB_DEPARTMENT
226 WHERE DEPARTMENT_NAME = '서반아어학과';
227
228 SELECT STUDENT_NAME AS "학생이름",
229        NVL(PROFESSOR_NAME, '지도교수 미지정') AS "지도교수"
230 FROM TB_STUDENT S
231 LEFT JOIN TB_PROFESSOR P ON (S.COACH_PROFESSOR_NO =
232 WHERE S.DEPARTMENT_NO = ( SELECT DEPARTMENT_NO
233        FROM TB_DEPARTMENT
234        WHERE DEPARTMENT_NAME = '서반아어학과' )
235 ORDER BY ENTRANCE_DATE;
236
237
238
239 /*15. 휴학생이 아닌 학생 중 평점이 4.0 이상인 학생을 찾아 그 학생의 학번, 이름, 학과
240 이름, 평점을 출력하는 SQL 문을 작성하시오.*/
241
242 /*강사님 풀이*/

```

```

243
244 SELECT STUDENT_NO AS "학번",
245 STUDENT_NAME AS "이름",
246 DEPARTMENT_NAME AS "학과 이름",
247 AVG (POINT)
248 FROM TB_STUDENT S
249 JOIN TB_DEPARTMENT D USING (DEPARTMENT_NO)
250 JOIN TB_GRADE G USING (STUDENT_NO)
251 WHERE ABSENCE_YN = 'N'
252 GROUP BY STUDENT_NO, STUDENT_NAME, DEPARTMENT_NAME
253 HAVING AVG (POINT) >= 4.0
254 ORDER BY STUDENT_NO;
255
256
257 /*16. 환경조경학과 전공과목들의 과목 별 평점을 파악할 수 있는 SQL 문을 작성하시오.*/
258 /*다시 풀기*/
259
260
261 SELECT DEPARTMENT_NO
262 FROM TB_DEPARTMENT
263 WHERE DEPARTMENT_NAME = '환경조경학과';
264 /*환경조경학과 DEPARTMENT_NO = 034*/
265
266 SELECT *
267 FROM TB_CLASS
268 JOIN TB_DEPARTMENT USING (DEPARTMENT_NO)
269 WHERE DEPARTMENT_NO = '034' OR CLASS_TYPE LIKE '전공%';
270
271
272 SELECT C.CLASS_NO, C.CLASS_NAME, AVG (POINT)
273 FROM TB_DEPARTMENT D
274 JOIN TB_CLASS C ON (D.DEPARTMENT_NO = C.DEPARTMENT_NO)
275 JOIN TB_GRADE G ON (C.CLASS_NO = G.CLASS_NO)
276 GROUP BY D.DEPARTMENT_NAME, C.CLASS_NO, C.CLASS_NAME,
277 HAVING D.DEPARTMENT_NAME = '환경조경학과' AND CLASS_TYPE LIKE
278 ORDER BY CLASS_NO;
279
280 /*17. 춘 기술대학교에 다니고 있는 최경희 학생과 같은 과 학생들의 이름과 주소를 출력하는
281 SQL 문을 작성하시오*/
282
283 SELECT DEPARTMENT_NO
284 FROM TB_STUDENT
285 WHERE STUDENT_NAME = '최경희'
286
287 SELECT STUDENT_NAME, STUDENT_ADDRESS
288 FROM TB_STUDENT
289 WHERE DEPARTMENT_NO = ( SELECT DEPARTMENT_NO
290 FROM TB_STUDENT

```

```

291 WHERE STUDENT_NAME = '최경희' );
292
293 /*18. 국어국문학과에서 총 평점이 가장 높은 학생의 이름과 학번을 표시하는 SQL 문을
294 작성하시오.*/
295
296 SELECT DEPARTMENT_NO
297 FROM TB_DEPARTMENT
298 WHERE DEPARTMENT_NAME = '국어국문학과';
299 /*국어국문학과 DEPARTMENT_NO = 001 */
300
301 SELECT S.DEPARTMENT_NO , AVG(POINT)
302 FROM TB_GRADE G
303 JOIN TB_STUDENT S ON (S.STUDENT_NO = G.STUDENT_NO)
304 GROUP BY S.STUDENT_NO, S.DEPARTMENT_NO
305 HAVING S.DEPARTMENT_NO = ( SELECT DEPARTMENT_NO
306 FROM TB_DEPARTMENT
307 WHERE DEPARTMENT_NAME = '국어국문학과' );
308
309 SELECT *
310 FROM TB_CLASS
311 WHERE DEPARTMENT_NO = '001';
312
313 SELECT STUDENT_NAME, STUDENT_SSN
314 FROM TB_STUDENT
315
316
317 /*19. 춘 기술대학교의 "환경조경학과"가 속한 같은 계열 학과들의 학과 별 전공과목 평점을
318 파악하기 위한 적절한 SQL 문을 찾아내시오. 단, 출력헤더는 "계열 학과명",
319 "전공평점"으로 표시되도록 하고, 평점은 소수점 한 자리까지만 반올림하여 표시되도록
320 한다.*/
321
322
323 SELECT CATEGORY
324 FROM TB_DEPARTMENT
325 WHERE DEPARTMENT_NAME = '환경조경학과';
326 /*자연과학 계열*/
327
328
329
330 SELECT DEPARTMENT_NAME AS "계열 학과명",
331 ROUND(AVG(POINT), 1) AS "전공평점"
332 FROM TB_DEPARTMENT D
333 JOIN TB_CLASS C USING (DEPARTMENT_NO)
334 JOIN TB_GRADE G USING (CLASS_NO)
335 WHERE CATEGORY = (SELECT CATEGORY
336 FROM TB_DEPARTMENT
337 WHERE DEPARTMENT_NAME = '환경조경학과')
338 GROUP BY DEPARTMENT_NAME

```

339<--\ 05 03 06
ORDER BY 1;

340

341

342

343

344

345

346

347

348

349

350

351

352

000514 08 0701 0001 0# 4-00